Ref #	Hits	Search Query	DBs	Default Operator	Plurals	Time Stamp
L4	21	(US-20040194070-$ or US-20030093776-$).did. or (US-4638423-$ or US-5301302-$ or US-5546552-$ or US-5560013-$ or US-5577231-$ or US-5577233-$ or US-5751982-$ or US-5790825-$ or US-5933622-$ or US-6009261-$ or US-6075937-$ or US-6142682-$ or US-6243668-$ or US-6516295-$ or US-6704925-$ or US-6785801-$ or US-5819063-$ or US-6163764-$ or US-4794522-$).did.	US-PGPUB; USPAT	OR	OFF	2005/06/24 18:04
S98	10	S89 and (instruction with (size length))	US-PGPUB; USPAT	OR	OFF	2005/06/24 14:53
S97	8	S89 and operand	US-PGPUB; USPAT	OR	OFF	2005/06/24 14:53
S96	11	S89 and byte	US-PGPUB; USPAT	OR	OFF	2005/06/24 14:52
S95	4	S89 and arrangement	US-PGPUB; USPAT	OR	OFF	2005/06/24 14:48
S89	20	(US-20040194070-$).did. or (US-4638423-$ or US-5301302-$ or US-5546552-$ or US-5560013-$ or US-5577231-$ or US-5577233-$ or US-5751982-$ or US-5790825-$ or US-5933622-$ or US-6009261-$ or US-6075937-$ or US-6142682-$ or US-6243668-$ or US-6516295-$ or US-6704925-$ or US-6785801-$ or US-5819063-$ or US-6163764-$ or US-4794522-$).did.	US-PGPUB; USPAT	OR	OFF	2005/06/24 14:48
S77	18	(US-20040194070-$).did. or (US-4638423-$ or US-5301302-$ or US-5546552-$ or US-5560013-$ or US-5577231-$ or US-5577233-$ or US-5751982-$ or US-5790825-$ or US-5933622-$ or US-6009261-$ or US-6075937-$ or US-6142682-$ or US-6243668-$ or US-6516295-$ or US-6704925-$ or US-6785801-$ or US-5819063-$).did.	US-PGPUB; USPAT	OR	OFF	2005/06/24 14:29
S88	50	("5539901" "5566121" "5566326" "5598553" "5604864" "5615328" "5630052" "5636227" "5644755" "5682481" "5694582" "5710934" "5720015" "5732201" "5749094" "5764659" "5765206" "5774694" "5787493" "5793714" "5796984" "5796566" "5802052" "5805473" "5815686" "5819015" "5822784" "5832299" "5842011" "5852720" "5857074" "5862083" "5867096" "5896393" "5910876" "5913052" "5940850" "5965860" "5973964" "5982371" "5983309" "6049866" "6052524" "6052383" "6055651" "6063131" "6070224" "6078520" "6106565" "6115813").pn.	USPAT	OR	OFF	2005/06/24 12:04
S87	48	("5918056" "6052685" "6240417" "4954942" "5278962" "5392420" "5623673" "5682310" "5970237" "6128732" "6212614" "6212614" "6397242" "6397379" "6446094" "5642491" "5815727" "5937185" "5953520" "5325512" "5590342" "5706407" "5909696" "6223284" "6223284" "4591967" "4611286" "4794522" "4812981" "4851828" "4888688" "4954968" "4975872" "5063499" "5056013" "5226154" "5278961" "5289581" "5289587" "5325469" "5357628" "5369749" "5369767" "5390314" "5438674" "5440710" "5452454" "5455926" "5485614" "5530673").pn.	USPAT	OR	OFF	2005/06/24 12:04
S85	146	717/138.ccls.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2005/06/24 12:03
S83	60	suspend with translation	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2005/06/24 11:43

S82	2	translation same (operand adj setting)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2005/06/24 11:38
S81	71	S69 and (store with instruction)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2005/06/24 11:16
S69	230	(instruction adj set adj simulat$4)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2005/06/24 11:14
S80	1	binary with translat$4 with (indirect in-direct) with address$4	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2005/06/24 11:12
S79	146	translat$4 with (indirect in-direct) with address$4	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2005/06/24 11:12
S78	9	S77 and stream	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2005/06/24 11:10
S76	15	resume adj translation	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2005/06/24 10:34
S75	0	resume adj tranlation	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2005/06/24 10:34
S74	229	S73 and (simulat$4 emulat$4 model$4)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2005/06/24 10:33
S73	379	S70 and S72	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2005/06/24 10:31
S72	2648	((in-direct indirect) adj address$4)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2005/06/24 10:30

S71	18	S69 and S70	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2005/06/24 10:27
S70	8291	(instruction byte operand) with align$8	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2005/06/24 10:17
S68	1	"09/992137"	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2005/06/24 10:16
S67	36	("5560013").URPN.	USPAT	OR	OFF	2005/06/24 09:59
S66	23	S64 and S65	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2005/06/24 09:53
S65	33	S63 and (emulation (instruction adj set adj simulat$4))	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2005/06/24 09:53
S64	61	S62 and (store with instruction) and (execution with (suspen$6 resum$5 stop$4 start$4))	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2005/06/24 09:52
S63	168	S62 and (store with instruction)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2005/06/24 09:51
S62	802	"S/390"	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2005/06/24 09:51
S2	798	"S/390"	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2005/06/24 09:51
S61	14	(legacy with instruction) and (emulat$4) and (execution with (suspen$6 resum$5 stop$4 start$4))	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2005/06/24 09:35

S60	17	(US-20040194070-$).did. or (US-4638423-$ or US-5301302-$ or US-5546552-$ or US-5560013-$ or US-5577233-$ or US-5751982-$ or US-5790825-$ or US-5933622-$ or US-6009261-$ or US-6075937-$ or US-6142682-$ or US-6516295-$ or US-6704925-$ or US-6785801-$ or US-6243668-$ or US-5577231-$).did.	US-PGPUB; USPAT	OR	OFF	2005/06/23 18:27
S59	2	(dynamic adj object adj code adj translation).ti.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2005/06/23 13:52
S58	15	S57 and modifi$6	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2005/06/22 21:09
S57	17	(US-20040194070-$).did. or (US-4638423-$ or US-5301302-$ or US-5546552-$ or US-5560013-$ or US-5577233-$ or US-5751982-$ or US-5790825-$ or US-5933622-$ or US-6009261-$ or US-6075937-$ or US-6142682-$ or US-6516295-$ or US-6704925-$ or US-6785801-$ or US-6243668-$ or US-5577231-$).did.	US-PGPUB; USPAT	OR	OFF	2005/06/22 21:08
S56	194	S55 and (TLB with (size index))	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2005/06/22 20:32
S55	1206	(instruction with translation) and (TLB)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2005/06/22 20:32
S54	1	(instruction with translation) and (block adj tracking adj table)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2005/06/22 20:32
S11	1	"09/992130"	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2005/06/22 20:31
S53	17126	fujitsu.as.	USPAT	OR	OFF	2005/06/22 16:24
S52	190	amdahl.as.	USPAT	OR	OFF	2005/06/22 16:24
S51	7	(instruction with translat$5) and hotspot	USPAT	OR	OFF	2005/06/22 16:18
S50	1	("6516295").URPN.	USPAT	OR	OFF	2005/06/22 16:11
S49	23	legacy with instruction with translation	US-PGPUB; USPAT; USOCR	OR	OFF	2005/06/22 16:01
S48	110	translation adj index$5	US-PGPUB; USPAT; USOCR	OR	OFF	2005/06/22 16:00
S47	61	S16 and (translation with (flag set indicator))	US-PGPUB; USPAT; USOCR	OR	OFF	2005/06/22 15:58

S46	5	S16 and (translation with (done complet$4) with (flag set indicator))	US-PGPUB; USPAT; USOCR	OR	OFF	2005/06/22 15:50
S16	715	S7 or S9	US-PGPUB; USPAT; USOCR	OR	OFF	2005/06/22 15:48
S45	82	S44 and S41	US-PGPUB; USPAT; USOCR	OR	OFF	2005/06/22 15:44
S44	581	(dynamic with translation) and index$5	US-PGPUB; USPAT; USOCR	OR	OFF	2005/06/22 15:44
S43	25	S15 and S41	US-PGPUB; USPAT; USOCR	OR	OFF	2005/06/22 15:43
S15	206	instruction adj set adj simulat$4	US-PGPUB; USPAT; USOCR	OR	OFF	2005/06/22 15:40
S42	172	((instruction with translation) and (index$5 with (block table) with translation))	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2005/06/22 15:31
S41	1192	((instruction with translation) and (block with translation))	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2005/06/22 15:31
S37	2551	(instruction and (block with translation))	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2005/06/22 15:30
S40	119	S39 and index with table	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2005/06/22 15:24
S39	470	S38 and table	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2005/06/22 15:24
S38	545	S37 and emulat$4	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2005/06/22 15:24
S7	317	(703/26).CCLS.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2005/06/22 15:22
S28	214	(instruction with translat$5 with (index flag table)) and (emulat$4)	US-PGPUB; USPAT; USOCR	OR	OFF	2005/06/22 14:26

S27	366	(instruction with translat$5 with (index flag table)) and (emulat$4 simulat$4 model$4)	US-PGPUB; USPAT; USOCR	OR	OFF	2005/06/22 14:03
S26	861	(instruction with translat$5 with (index flag table set)) and (emulat$4 simulat$4 model$4)	US-PGPUB; USPAT; USOCR	OR	OFF	2005/06/22 14:03
S25	18148	(translat$5 with (index flag table set)) and (emulat$4 simulat$4 model$4)	US-PGPUB; USPAT; USOCR	OR	OFF	2005/06/22 13:33
S21	190	S16 and flag	US-PGPUB; USPAT; USOCR	OR	OFF	2005/06/22 13:31
S23	1	S16 and (block with transform)	US-PGPUB; USPAT; USOCR	OR	OFF	2005/06/22 13:27
S24	1	S23 and address	US-PGPUB; USPAT; USOCR	OR	OFF	2005/06/22 13:26
S19	63	S16 and (table with index)	US-PGPUB; USPAT; USOCR	OR	OFF	2005/06/22 13:26
S22	11	S16 and translation with flag	US-PGPUB; USPAT; USOCR	OR	OFF	2005/06/22 13:13
S20	15	S16 and ((table with index) same translat$5)	US-PGPUB; USPAT; USOCR	OR	OFF	2005/06/22 13:13
S18	245	S17 and (translat$5)	US-PGPUB; USPAT; USOCR	OR	OFF	2005/06/22 12:24
S17	433	S16 and (table or index)	US-PGPUB; USPAT; USOCR	OR	OFF	2005/06/22 12:24
S14	9	("4574344" | "4635188" | "4638423" | "4761733" | "5333287" | "5406644" | "5430862" | "5481693" | "5546552").PN.	US-PGPUB; USPAT; USOCR	OR	OFF	2005/06/22 11:45
S13	18	S12 and (store with instruction)	USPAT	OR	OFF	2005/06/22 10:30
S12	33	("4638423").URPN.	USPAT	OR	OFF	2005/06/22 10:29
S8	82	S7 and (instruction with translat$)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2005/06/22 10:22
S10	58	S9 and (instruction with translat$)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2005/06/22 10:17
S9	484	703/27.ccls.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2005/06/22 10:17
S6	2	("5313614" | "5404478").PN.	US-PGPUB; USPAT; USOCR	OR	OFF	2005/06/21 12:05

S5	19	"S/390" with emulat$4	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2005/06/21 12:05
S4	116	S2 and emulat$4	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2005/06/21 12:04
S3	4	"S/390" with legacy with instruction	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2005/06/21 12:04
S1	165	legacy with instruction with (translation emulat$4 simulat$4 execut$4)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2005/06/21 12:03

Terms used binary translation byte alignment Found 225 of 157,908

Sort results by relevance | Save results to a Binder | Try an Advanced Search
Display results expanded form | Search Tips | Try this search in The ACM Guide
Open results in a new window

Results 1 - 20 of 200 Result page: 1 2 3 4 5 6 7 8 9 10 next
Best 200 shown Relevance scale

1 Native code compilation of Erlang's bit syntax
Per Gustafsson, Konstantinos Sagonas
October 2002 **Proceedings of the 2002 ACM SIGPLAN workshop on Erlang**
Full text available: pdf(126.61 KB) Additional Information: full citation, abstract, references, citings

Erlang's bit syntax caters for flexible pattern matching on bit streams (objects known as *binaries*). Binaries are nowadays heavily used in typical Erlang applications such as protocol programming, which in turn has created a need for efficient support of the basic operations on binaries.To this effect, we describe a scheme for efficient native code compilation of Erlang's bit syntax. The scheme relies on *partial translation* for avoiding code explosion, an ...

2 Binary translation and architecture convergence issues for IBM system/390
Michael Gschwind, Kemal Ebcioğlu, Erik Altman, Sumedh Sathaye
May 2000 **Proceedings of the 14th international conference on Supercomputing**
Full text available: pdf(1.44 MB) Additional Information: full citation, abstract, references, index terms

We describe the design issues in an implementation of the ESA/390 architecture based on binary translation to a very long instruction word (VLIW) processor. During binary translation, complex ESA/390 instructions are decomposed into instruction "primitives" which are then scheduled onto a wide-issue machine. The aim is to achieve high instruction level parallelism due to the increased scheduling and optimization opportunities which can be exploited by binary translation software ...

3 Machine-adaptable dynamic binary translation
David Ung, Cristina Cifuentes
January 2000 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN workshop on Dynamic and adaptive compilation and optimization**, Volume 35 Issue 7
Full text available: pdf(1.72 MB) Additional Information: full citation, abstract, references, citings, index terms

Dynamic binary translation is the process of translating and optimizing executable code for one machine to another at runtime, while the program is "executing" on the target machine.

Dynamic translation techniques have normally been limited to two particular machines; a competitor's machine and the hardware manufacturer's machine. This research provides for a more general framework for dynamic translations, by providing a framework based on specifications of machines that ...

Keywords: binary translation, dynamic compilation, dynamic execution, emulation, interpretation

4 Binary translation
Richard L. Sites, Anton Chernoff, Matthew B. Kirk, Maurice P. Marks, Scott G. Robinson
February 1993 **Communications of the ACM**, Volume 36 Issue 2
Full text available: pdf(4.64 MB) Additional Information: full citation, references, citings, index terms

Keywords: CISC computers, RISC computers, binary translation, computer architecture, processor architecture translation

5 Data translation: DSCL: a Data Specification and Conversion Language for networks

G. Michael Schneider
May 1975 **Proceedings of the 1975 ACM SIGMOD international conference on Management of data**
Full text available: pdf(1.69 MB) Additional Information: full citation, abstract, references, citings

The rapid growth of large, heterogeneous, resource-sharing computer networks has created a serious problem in the sharing of information between incompatible systems. These incompatibilities can be categorized as either physical or logical in nature. Physical incompatibilities are problems caused by the way that the individual binary digits, regardless of what information they represent, are generated or stored internally. This would include character, word, and record size differences, blocking ...

6 Using Dynamic Binary Translation to Fuse Dependent Instructions
Shiliang Hu, James E. Smith
March 2004 **Proceedings of the international symposium on Code generation and optimization: feedback-directed and runtime optimization**
Full text available: pdf(240.60 KB) Additional Information: full citation, abstract, index terms

Instruction scheduling hardware can be simplifiedand easily pipelined if pairs of dependent instructionsare fused so they share a single instruction schedulingslot. We study an implementation of the x86 ISA thatdynamically translates x86 code to an underlying ISAthat supports instruction fusing. A microarchitecturethat is co-designed with the fused instruction set completesthe implementation.In this paper, we focus on the dynamic binarytranslator for such a co-designed x86 virtual machine.The dy ...

7 A high-performance object-oriented memory
Craig Hyatt
September 1993 **ACM SIGARCH Computer Architecture News**, Volume 21 Issue 4
Full text available: pdf(819.75 KB) Additional Information: full citation, abstract, index terms

The proposed design places a high-performance object memory in a portable peripheral that relieves the host CPU of the burden of object address translation and the constant management of live objects. This flexible approach is designed for the newest generation of high-performance workstations running 32-bit Smalltalk, Lisp, C++ and other OOP (object oriented programming) environments. With a two-layer cache and 50ns DRAM object memory, the hardware is capable of accessing object data (including ...

Keywords: RISC, binary-buddy allocation, capability-based protection

8 Wisconsin Architectural Research Tool Set
Mark D. Hill, James R. Larus, Alvin R. Lebeck, Madhusudhan Talluri, David A. Wood
September 1993 **ACM SIGARCH Computer Architecture News**, Volume 21 Issue 4
Full text available: pdf(1.16 MB) Additional Information: full citation, citings, index terms

9 Revised Report of the Algorithmic Language Algol 68
Peter Naur
January 1978 **ALGOL Bulletin**, Issue Sup 47
Full text available: pdf(4.70 MB) Additional Information: full citation, index terms

10 Language-based security: SELF: a transparent security extension for ELF binaries
Daniel C. DuVarney, V. N. Venkatakrishnan, Sandeep Bhatkar
August 2003 **Proceedings of the 2003 workshop on New security paradigms**
Full text available: pdf(1.03 MB) Additional Information: full citation, abstract, references

The ability to analyze and modify binaries is often very useful from a security viewpoint. Security operations one would like to perform on binaries include the ability to extract models of program behavior and insert inline reference monitors. Unfortunately, the existing manner in which binary code is packaged prevents even the simplest of analyses, such as distinguishing code from data, from succeeding 100 percent of the time. In this paper, we propose SELF, a security-enhanced ELF (Executable ...

11 A dynamic binary translation approach to architectural simulation
Harold W. Cain, Kevin M. Lepak, Mikko H. Lipasti
March 2001 **ACM SIGARCH Computer Architecture News**, Volume 29 Issue 1
Full text available: pdf(618.63 KB) Additional Information: full citation, abstract, index terms

We present the design of a PowerPC-based simulation infrastructure for architectural research. Our infrastructure uses an execution-driven out-of-order processor timing simulator from the SimpleScalar tool set. While porting SimpleScalar to the PowerPC architecture, we would like to remain compatible

with other versions of SimpleScalar. We accomplish this by performing dynamic binary translation of the PowerPC instruction set architecture to the SimpleScalar instruction set architecture, and by ...

12 Emerging applications: Obfuscation of executable code to improve resistance to static disassembly
Cullen Linn, Saumya Debray
October 2003 **Proceedings of the 10th ACM conference on Computer and communications security**

Full text available: pdf(142.75 KB) Additional Information: full citation, abstract, references, citings, index terms

A great deal of software is distributed in the form of executable code. The ability to reverse engineer such executables can create opportunities for theft of intellectual property via software piracy, as well as security breaches by allowing attackers to discover vulnerabilities in an application. The process of reverse engineering an executable program typically begins with disassembly, which translates machine code to assembly code. This is then followed by various decompilation steps that ai ...

Keywords: code obfuscation, disassembly

13 A language-based design for portable data files
C. Burch
May 1989 **ACM SIGPLAN Notices**, Volume 24 Issue 5

Full text available: pdf(350.20 KB) Additional Information: full citation, abstract, index terms

Currently data files to be accessed remotely from dissimilar systems must be transformed to the local language processors' file format and data representation; a process that has changed little since punch cards were the main form of portable data files. A standard is proposed for languages to use the data type information available to the runtime library to make these data transformations before the data is transferred to the file or the user's variables. By specifying one binary format for eac ...

14 Draft Proposed: American National Standard—Graphical Kernel System
Technical Committee X3H3 - Computer Graphics
February 1984 **ACM SIGGRAPH Computer Graphics**, Volume 18 Issue SI

Full text available: pdf(16.07 MB) Additional Information: full citation

15 Vectorization for SIMD architectures with alignment constraints
Alexandre E. Eichenberger, Peng Wu, Kevin O'Brien
June 2004 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 2004 conference on Programming language design and implementation**, Volume 39 Issue 6

Full text available: pdf(241.50 KB) Additional Information: full citation, abstract, references, index terms

When vectorizing for SIMD architectures that are commonly employed by today's multimedia extensions, one of the new challenges that arise is the handling of memory alignment. Prior research has focused primarily on vectorizing loops where all memory references are properly aligned. An important aspect of this problem, namely, how to vectorize misaligned memory references, still remains unaddressed.This paper presents a compilation scheme that systematically vectorizes loops in the presence of mi ...

Keywords: SIMD, alignment, compiler, multimedia extensions, simdization, vectorization

16 A C language extension for machine-independent programming
Shingo Kamiya, Toshiyuki Yoshida, Takanobu Sugiyasu, Koki Miyazawa
December 1986 **Proceedings of the 1986 ACM SIGSMALL/PC symposium on Small systems**

Full text available: pdf(759.20 KB) Additional Information: full citation, abstract, references, index terms

MIC (Machine-Independent C) is an extension of the C language which has been designed to write portable programs as installed in various small computers. MIC provides unified semantics suitable for typical small computers with new facilities for machine-independent data definition, and its syntax conforms to the preliminary draft of the proposed ANSI standard for C. It is fully implemented as a compiler front end called MICP, and has been applied to actual programming. The principal feature ...

17 Compiler transformations for high-performance computing
David F. Bacon, Susan L. Graham, Oliver J. Sharp
December 1994 **ACM Computing Surveys (CSUR)**, Volume 26 Issue 4

Full text available: pdf(6.32 MB) Additional Information: full citation, abstract, references, citings, index terms, review

In the last three decades a large number of compiler transformations for optimizing programs have been implemented. Most optimizations for uniprocessors reduce the number of instructions executed by the program using transformations based on the analysis of scalar quantities and data-flow techniques. In contrast, optimizations for high-performance superscalar, vector, and parallel processors maximize

parallelism and memory locality with transformations that rely on tracking the properties o ...

Keywords: compilation, dependence analysis, locality, multiprocessors, optimization, parallelism, superscalar processors, vectorization

18 Efficient wire formats for high performance computing
Fabian E. Bustamante, Greg Eisenhauer, Karsten Schwan, Patrick Widener
November 2000 **Proceedings of the 2000 ACM/IEEE conference on Supercomputing (CDROM)**

Full text available: pdf(134.14 KB) Publisher Site Additional Information: full citation, abstract, references, citings, index terms

High performance computing is being increasingly utilized in non-traditional circumstances where it must interoperate with other applications. For example, online visualization is being used to monitor the progress of applications, and real-world sensors are used as inputs to simulations. Whenever these situations arise, there is a question of what communications infrastructure should be used to link the different components. Traditional HPC-style communications systems such as MPI offer r ...

19 Retrospective: what have we learned from the PDP-11—what we have learned from VAX and Alpha
Gorden Bell, W. D. Strecker
August 1998 **25 years of the international symposia on Computer architecture (selected papers)**

Full text available: pdf(816.63 KB) Additional Information: full citation, references, index terms

20 Randomized instruction set emulation
Elena Gabriela Barrantes, David H. Ackley, Stephanie Forrest, Darko Stefanović
February 2005 **ACM Transactions on Information and System Security (TISSEC)**, Volume 8 Issue 1

Full text available: pdf(374.44 KB) Additional Information: full citation, abstract, references, index terms

Injecting binary code into a running program is a common form of attack. Most defenses employ a "guard the doors" approach, blocking known mechanisms of code injection. *Randomized instruction set emulation* (RISE) is a complementary method of defense, one that performs a hidden randomization of an application's machine code. If foreign binary code is injected into a program running under RISE, it will not be executable because it will not know the proper randomization. The pape ...

Keywords: Automated diversity, randomized instruction sets, software diversity

Results 1 - 20 of 200 Result page: 1 2 3 4 5 6 7 8 9 10 next

Terms used binary translation indirect address Found 920 of 197,998

Sort results by relevance Save results to a Binder Try an Advanced Search
Display results expanded form Search Tips Try this search in The ACM Guide
Open results in a new window

Results 1 - 20 of 200 Result page: 1 2 3 4 5 6 7 8 9 10 next
Best 200 shown Relevance scale

1 Hardware Support for Control Transfers in Code Caches
Ho-Seop Kim, James E. Smith
December 2003 **Proceedings of the 36th annual IEEE/ACM International Symposium on Microarchitecture**
Full text available: pdf(115.74 KB) Additional information: full citation, abstract, cited by, index terms

Many dynamic optimization and/or binary translationsystems hold optimized/translated superblocks in a codecache. Conventional code caching systems suffer fromoverheads when control is transferred from one cachedsuperblock to another, especially via register-indirectjumps. The basic problem is that instruction addresses inthe code cache are different from those in the original programbinary. Therefore, performance for register-indirectjumps depends on the ability to translate efficiently fromsour ...

2 Binary translation and architecture convergence issues for IBM system/390
Michael Gschwind, Kemal Ebcioğlu, Erik Altman, Sumedh Sathaye
May 2000 **Proceedings of the 14th international conference on Supercomputing**
Full text available: pdf(1.44 MB) Additional information: full citation, abstract, references, index terms

We describe the design issues in an implementation of the ESA/390 architecture based on binary translation to a very long instruction word (VLIW) processor. During binary translation, complex ESA/390 instructions are decomposed into instruction "primitives" which are then scheduled onto a wide-issue machine. The aim is to achieve high instruction level parallelism due to the increased scheduling and optimization opportunities which can be exploited by binary translation software ...

3 Hardware and Binary Modification Support for Code Pointer Protection From Buffer Overflow
Nathan Tuck, Brad Calder, George Varghese
December 2004 **Proceedings of the 37th annual International Symposium on Microarchitecture**
Full text available: pdf(274.15 KB) Additional information: full citation, abstract

Buffer overflow vulnerabilities are currently the most prevalent security vulnerability; they are responsible for over half of the CERT advisories issued in the last three years.Since many attacks exploit buffer overflow vulnerabilities, techniques that prevent buffer overflow attacks would greatly increase the difficulty of writing a new worm. This paper examines both software and hardware solutions for protecting code pointers from buffer overflow attacks.We first evaluate the performance over ...

4 Communications between independently translated blocks
Peter Wegner
July 1962 **Communications of the ACM**, Volume 5 Issue 7
Full text available: pdf(531.77 KB) Additional information: full citation, references

5 Dynamic translation: Dynamic binary translation for accumulator-oriented architectures
Ho-Seop Kim, James E. Smith
March 2003 **Proceedings of the international symposium on Code generation and optimization: feedback-directed and runtime optimization**
Full text available: pdf(1.12 MB), Publisher Site Additional information: full citation, abstract, citations, index terms

A dynamic binary translation system for a co-designed virtual machine is described and evaluated. The underlying hardware directly executes an accumulator-oriented instruction set that exposes instruction dependence chains (strands) to a distributed microarchitecture containing a simple instruction pipeline.

To support conventional program binaries, a source instruction set (Alpha in our study) is dynamically translated to the target accumulator instruction set. The binary translator identifies ...

6 Using Dynamic Binary Translation to Fuse Dependent Instructions
Shiliang Hu, James E. Smith
March 2004 **Proceedings of the international symposium on Code generation and optimization: feedback-directed and runtime optimization**
Full text available: pdf(260.50 KB) Additional information: full citation, abstract, index terms

Instruction scheduling hardware can be simplifiedand easily pipelined if pairs of dependent instructionsare fused so they share a single instruction schedulingslot. We study an implementation of the x86 ISA thatdynamically translates x86 code to an underlying ISAthat supports instruction fusing. A microarchitecturethat is co-designed with the fused instruction set completesthe implementation.In this paper, we focus on the dynamic binarytranslator for such a co-designed x86 virtual machine.The dy ...

7 Generic control flow reconstruction from assembly code
Daniel Kästner, Stephan Wilhelm
June 2002 **ACM SIGPLAN Notices , Proceedings of the joint conference on Languages, compilers and tools for embedded systems: software and compilers for embedded systems**, Volume 37 Issue 7
Full text available: pdf(250.08 KB) Additional information: full citation, abstract, references, citings, index terms

Processors used in embedded systems are usually characterized by specialized irregular hardware architectures for which traditional code generation and optimization techniques fail. Especially for these types of processors the Propan system has been developed that enables high-quality machine-dependent postpass optimizers to be generated from a concise hardware specification. Optimizing code transformations as featured by Propan require the control flow graph of the input progr ...

Keywords: assembly code, call graph, control flow reconstruction, embedded processors, postpass optimization, retargetable compilers

8 Emerging applications: Obfuscation of executable code to improve resistance to static disassembly
Cullen Linn, Saumya Debray
October 2003 **Proceedings of the 10th ACM conference on Computer and communications security**
Full text available: pdf(315.75 KB) Additional information: full citation, abstract, references, citings, index terms

A great deal of software is distributed in the form of executable code. The ability to reverse engineer such executables can create opportunities for theft of intellectual property via software piracy, as well as security breaches by allowing attackers to discover vulnerabilities in an application. The process of reverse engineering an executable program typically begins with disassembly, which translates machine code to assembly code. This is then followed by various decompilation steps that al ...

Keywords: code obfuscation, disassembly

9 Machine-adaptable dynamic binary translation
David Ung, Cristina Cifuentes
January 2000 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN workshop on Dynamic and adaptive compilation and optimization**, Volume 35 Issue 7
Full text available: pdf(1.21 MB) Additional information: full citation, abstract, references, citings, index terms

Dynamic binary translation is the process of translating and optimizing executable code for one machine to another at runtime, while the program is "executing" on the target machine.

Dynamic translation techniques have normally been limited to two particular machines; a competitor's machine and the hardware manufacturer's machine. This research provides for a more general framework for dynamic translations, by providing a framework based on specifications of machines that ...

Keywords: binary translation, dynamic compilation, dynamic execution, emulation, interpretation

10 Dynamic analysis: The design and implementation of FIT: a flexible instrumentation toolkit
Bruno De Bus, Dominique Chanet, Bjorn De Sutter, Ludo Van Put, Koen De Bosschere
June 2004 **Proceedings of the ACM-SIGPLAN-SIGSOFT workshop on Program analysis for software tools and engineering**
Full text available: pdf(252.10 KB) Additional information: full citation, abstract, references, index terms

This paper presents FIT, a Flexible open-source binary code Instrumentation Toolkit. Unlike existing tools, FIT is truly portable, with existing backends for the Alpha, x86 and ARM architectures and the Tru64Unix, Linux and ARM Firmware execution environments. This paper focuses on some of the

problems that needed to be addressed for providing this degree of portability. It also discusses the trade-off between instrumentation precision and low overhead.

Keywords: code compaction, performance code abstraction

[11] Dynamic translation: Retargetable and reconfigurable software dynamic translation
K. Scott, N. Kumar, S. Velusamy, B. Childers, J. W. Davidson, M. L. Soffa
March 2003 **Proceedings of the international symposium on Code generation and optimization: feedback-directed and runtime optimization**

Full text available: pdf(1.14 MB) Publisher Site Additional Information: full citation, abstract, references, citings, index terms

Software dynamic translation (SDT) is a technology that permits the modification of an executing program's instructions. In recent years, SDT has received increased attention, from both industry and academia, as a feasible and effective approach to solving a variety of significant problems. Despite this increased attention, the task of initiating a new project in software dynamic translation remains a difficult one. To address this concern, and in particular, to promote the adoption of SDT techn ...

[12] Optimizations and oracle parallelism with dynamic translation
Kemal Ebcioğlu, Erik R. Altman, Michael Gschwind, Sumedh Sathaye
November 1999 **Proceedings of the 32nd annual ACM/IEEE international symposium on Microarchitecture**

Full text available: pdf(1.27 MB) Publisher Site Additional Information: full citation, abstract, references, citings, index terms

We describe several optimizations which can be employed in a dynamic binary translation (DBT) system, where low compilation/translation overhead is essential. These optimizations achieve a high degree of ILP, sometimes even surpassing a static compiler employing more sophisticated, and more time-consuming algorithms [9]. We present results in which we employ these optimizations in a dynamic binary translation system capable of computing oracle parallelism.

[13] RIFLE: An Architectural Framework for User-Centric Information-Flow Security
Neil Vachharajani, Matthew J. Bridges, Jonathan Chang, Ram Rangan, Guilherme Ottoni, Jason A. Blome, George A. Reis, Manish Vachharajani, David I. August
December 2004 **Proceedings of the 37th annual International Symposium on Microarchitecture**

Full text available: pdf(178.61 KB) Additional Information: full citation, abstract

Even as modern computing systems allow the manipulation and distribution of massive amounts of information, users of these systems are unable to manage the confidentiality of their data in a practical fashion. Conventional access control security mechanisms cannot prevent the illegitimate use of privileged data once access is granted. For example, information provided by a user during an online purchase may be covertly delivered to malicious third parties by an untrustworthy web browser. Existin ...

[14] An architectural framework for migration from CISC to higher performance platforms
Gabriel M. Silberman, Kemal Ebcioğlu
August 1992 **Proceedings of the 6th International conference on Supercomputing**

Full text available: pdf(2.04 MB) Additional Information: full citation, abstract, references, citings, index terms

We describe a novel architectural framework that allows software applications written for a given Complex Instruction Set Computer (CISC) to migrate to a different, higher performance architecture, without a significant investment on the part of the application user or developer. The framework provides a hardware mechanism for seamless switching between two instruction sets, resulting in a machine that enhances application performance while keeping the same program behavior (from a user per ...

[15] Migrating a CISC computer family onto RISC via object code translation
Kristy Andrews, Duane Sand
September 1992 **ACM SIGPLAN Notices , Proceedings of the fifth international conference on Architectural support for programming languages and operating systems**, Volume 27 Issue 9

Full text available: pdf(1.12 MB) Additional Information: full citation, references, citings, index terms

[16] An advanced tactical computer concept
Kenneth J. Thurber, Peter C. Patton, Robert C. Deward, Jon C. Strauss, Thomas W. Petschauer
March 1977 **ACM SIGARCH Computer Architecture News , Proceedings of the 4th annual symposium on Computer architecture**, Volume 5 Issue 7

Full text available: pdf(1.42 MB) Additional Information: full citation, abstract, references, index terms

This paper discusses the design of a real-time computer. The computer's design requirements, design decisions, and architecture are summarized. The paper discusses how the design requirements influenced the computer architecture. The system's three upward compatible addressing options (real, base, virtual) are also discussed.

[17] Dynamic translation: The Transmeta Code Morphing™ Software: using speculation, recovery, and adaptive retranslation to address real-life challenges
James C. Dehnert, Brian K. Grant, John P. Banning, Richard Johnson, Thomas Kistler, Alexander Klaiber, Jim Mattson
March 2003 **Proceedings of the international symposium on Code generation and optimization: feedback-directed and runtime optimization**

Full text available: pdf(913.75 KB) Publisher Site Additional Information: full citation, abstract, references, citings, index terms, review

Transmeta's Crusoe microprocessor is a full, system-level implementation of the x86 architecture, comprising a native VLIW microprocessor with a software layer, the **Code Morphing Software** (CMS), that combines an interpreter, dynamic binary translator, optimizer, and runtime system. In its general structure, CMS resembles other binary translation systems described in the literature, but it is unique in several respects. The wide range of PC workloads that CMS must handle gracefully in real ...

Keywords: binary translation, dynamic optimization, dynamic translation, emulation, self-modifying code, speculation

[18] Limitations on the portability of real time Ada programs
T. Griest, m. Bender
January 1988 **Proceedings of the conference on Tri-Ada '89: Ada technology in context: application, development, and deployment**

Full text available: pdf(1.76 MB) Additional Information: full citation, abstract, references, index terms

This paper describes areas of the Ada language where transportability is restricted by the fact that implementation choices are allowed. Transportability guidelines have been developed previously for Ada [2,3,10], but have taken the approach that only features of the language which are guaranteed to be supported by all implementations should be used. The "common intersection" approach rules out use of many features that were included in Ada because they are required for real-tim ...

[19] A search algorithm and data structure for an efficient information system
Shou-chuan Yang
September 1969 **Proceedings of the 1969 conference on Computational linguistics**

Full text available: pdf(1.04 MB) Additional Information: full citation, abstract, references

This paper describes a system for information storage, retrieval, and updating, with special attention to the search algorithm and data structure demanded for maximum program efficiency. The program efficiency is especially warranted when a natural language or a symbolic language is involved in the searching process.The system is a basic framework for an efficient information system. It can be implemented for text processing and document retrieval; numerical data retrieval; and for handling of la ...

[20] An underlay strategy for indirect routing
Aline Carneiro Viana, Marcelo Dias de Amorim, Serge Fdida, José Ferreira de Rezende
November 2004 **Wireless Networks**, Volume 10 Issue 6

Full text available: pdf(345.24 KB) Additional Information: full citation, abstract, references, index terms

The evolution of the Internet toward ubiquity, mobility, and independence of wired infrastructure requires revising routing in large dynamic clouds. The need for frequent address updates caused by node mobility suggests decoupling the permanent node identifier from its topological address. This paper proposes Tribe, an indirect and scalable routing protocol for self-organizing networks. Tribe provides an anchor-based abstraction, where the communication is split into two phases: location of t ...

Keywords: indirect routing, peer-to-peer communication, self-organization

Results 1 - 20 of 200 Result page: 1 2 3 4 5 6 7 8 9 10 next

Terms used MVC instruction Found 20,782 of 157,906

Sort results by relevance | Save results to a Binder | Try an Advanced Search
Display results expanded form | Search Tips | Try this search in The ACM Guide
Open results in a new window

Results 1 - 20 of 200 Result page: 1 2 3 4 5 6 7 8 9 10 next
Best 200 shown Relevance scale

1 An instruction timing model of CPU performance
Bernard L. Peuto, Leonard J. Shustek
March 1977 **ACM SIGARCH Computer Architecture News , Proceedings of the 4th annual symposium on Computer architecture,** Volume 5 Issue 7

A model of high-performance computers is derived from instruction timing formulas, with compensation for pipeline and cache memory effects. The model is used to predict the performance of the IBM 370/168 and the Amdahl 470 V/6 on specific programs,/and the results are verified by comparison with actual performance. Data collected about program behavior is combined with the performance analysis to highlight some of the problems with high-performance implementations of such architectures.

2 An instruction timing model of CPU performance
Bernard L. Peuto, Leonard J. Shustek
August 1988 **25 years of the international symposia on Computer architecture (selected papers)**

3 Analyzing exotic instructions for a retargetable code generator
Thomas M. Morgan, Lawrence A. Rowe
June 1982 **ACM SIGPLAN Notices , Proceedings of the 1982 SIGPLAN symposium on Compiler construction,** Volume 17 Issue 6

Exotic instructions are complex instructions, such as block move, string search, and string edit, which are found on most conventional computers. Recent retargetable code generator and instruction set analysis systems have not dealt with exotic instructions. A method to analyze exotic instructions is presented which provides the information needed by a retargetable code generator. The analysis uses source-to-source transformations to prove the equivalence of high-level language operators to ...

4 Assembler-Language Macroprogramming: A Tutorial Oriented Toward the IBM 360
William Kent
December 1968 **ACM Computing Surveys (CSUR),** Volume 1 Issue 4

5 Coding guidelines for pipelined processors
James W. Rymarczyk
March 1982 **Proceedings of the first international symposium on Architectural support for programming languages and operating systems,** Volume 10 , 17 Issue 2 , 4

This paper is a tutorial for assembly language programmers of pipelined processors. It describes the general characteristics of pipelined processors and presents a collection of coding guidelines for them. These guidelines are particularly significant to compiler developers who determine object code patterns.

6 A microcoded tool to sample the software instruction address
C. Edward Armbruster
November 1979 **ACM SIGMICRO Newsletter , Proceedings of the 12th annual workshop on Microprogramming,** Volume 10 Issue 4

This paper introduces the concept of software execution localization and defines a microcoded tool for measuring execution localization. The tool is presented as a conceptual description. This tool has been used during development of microcode assists for the System/370 Models 138 and 148, and also the 4341 Processor.

7 Instruction translation for an experimental S/390 processor
Rolf Hilgendorf, Wolfram Sauer
March 2001 **ACM SIGARCH Computer Architecture News,** Volume 29 Issue 1

The IBM™ S/390™ architecture is a complex architecture, which has grown over a long period of time. Typical implementations use microcode to cope with the more complex instructions and facilities of S/390. Current IBM S/390 processors even contain two levels of microcode.We report on an experimental S/390 processor based on a RISC processor kernel employing superscalar, out of order execution of instructions. S/390 instructions have to be translated into internal sequences of ...

8 Binary translation and architecture convergence issues for IBM system/390
Michael Gschwind, Kemal Ebcioğlu, Erik Altman, Sumedh Sathaye
May 2000 **Proceedings of the 14th international conference on Supercomputing**

We describe the design issues in an implementation of the ESA/390 architecture based on binary translation to a very long instruction word (VLIW) processor. During binary translation, complex ESA/390 instructions are decomposed into instruction "primitives" which are then scheduled onto a wide-issue machine. The aim is to achieve high instruction level parallelism due to the increased scheduling and optimization opportunities which can be exploited by binary translation software ...

9 Internet Nuggets
Mark Thorson
May 1988 **ACM SIGARCH Computer Architecture News,** Volume 26 Issue 2

This column consists of selected traffic from the *comp.arch* newsgroup, a forum for discussion of computer architecture on Internet---an international computer network.

10 Mimic: a fast system/370 simulator
C. May
July 1987 **ACM SIGPLAN Notices , Papers of the Symposium on Interpreters and interpretive techniques,** Volume 22 Issue 7

Software simulation of one computer on another tends to be slow. Traditional simulators typically execute about 100 instructions on the host machine per instruction simulated. Newer simulators reduce the expansion factor to about 10, by saving and reusing translations of individual instructions. This paper describes an experimental simulator which takes the progression one step further, translating groups of instructions as a unit. This approach, combined with flow analysis, reduces the expansio ...

11 Cobol vs. PL/1: some performance comparisons
Paul J. Jalics
March 1984 **Communications of the ACM,** Volume 27 Issue 3

12 PL/I program efficiency
Michael McNeil, William Tracz
June 1980 **ACM SIGPLAN Notices,** Volume 15 Issue 6

All PL/I Programmers should be aware of and genuinely concerned about PL/I Program efficiency. This paper addresses the following question:"How do you write a PL/I program which the PL/I Compiler will reduce to the smallest and fastest executing machine language module?" The real world payoffs of knowing how the PL/I Optizing Compiler handles different syntactical representations of similar semantic relationships with respect to code generation and storage allocation can increase program runtime ...

13 On the use of naming and binding in early courses
Mark Smotherman
February 1987 **ACM SIGCSE Bulletin , Proceedings of the eighteenth SIGCSE technical symposium on**

Computer science education, Volume 19 Issue 1

In most computer science curricula, the concepts of naming and binding are explicitly treated only in a small number of the later courses, such as operating systems and programming language foundations. However, these concepts are fundamental and underlie the whole of computer science. In this paper, a proposal is made to explicitly introduce these concepts in the second or third course so that they may be used in the analysis of ideas encountered throughout a student's program of study. Th ...

14 A case study of a new code generation technique for compilers
J. Lawrence Carter
December 1977 **Communications of the ACM**, Volume 20 Issue 12

Keywords: PL/I compiler, code generation, compiler structure, concatenation, data flow analysis, optimization techniques, optimizing compiler, program optimization

15 A study of interleaved memory systems by trace driven simulation
Fred W. Terman
July 1978 **ACM SIGSIM Simulation Digest , Proceedings of the 4th symposium on Simulation of computer systems**, Volume 7 Issue 4

A model of interleaved memory systems for IBM System/360 and System/370 architecture has been investigated by means of a trace driven simulation. The model used is an extension of one due to G. J. Burnett and E. G. Coffman, Jr. The trace data to drive the simulation was obtained from instruction-by-instruction traces of typical IBM 360/370 programs and of the OS/VS2 operating system. The predictions of the Burnett-Coffman model are found to fit well with the simulation results for the fetch ...

16 Emulation of large systems
S. G. Tucker
December 1965 **Communications of the ACM**, Volume 8 Issue 12

17 Cache Memories
Alan Jay Smith
September 1982 **ACM Computing Surveys (CSUR)**, Volume 14 Issue 3

18 Characterizing the Storage Process and Its Effect on the Update of Main Memory by Write Through
Alan Jay Smith
January 1979 **Journal of the ACM (JACM)**, Volume 26 Issue 1

19 Using squeak for teaching user interface software
Mark Guzdial
February 2001 **ACM SIGCSE Bulletin , Proceedings of the thirty-second SIGCSE technical symposium on Computer Science Education**, Volume 33 Issue 1

Squeak is a new programming language that is particularly appropriate for learning computer science. It offers an excellent infrastructure for interesting projects (e.g., multimedia, Web browsing and serving), and all source code is included (and written in Squeak) from the virtual machine, windowing, on up. Squeak is being used in a course on *Objects and Design* (focusing on the development of user interfaces), both to enhance the infrastructure for a course on, and to change how user int ...

20 Control store implementation of a high performance VLSI CISC
J. H. Chang, H. H. Chao, K. Lewis, M. Holland
January 1988 **Proceedings of the 21st annual workshop on Microprogramming and microarchitecture**

The implementation of the &mgr;-sequencer and the large loadable control store of a high performance CMOS 370 system [1] is described. The control store consists of two parts, a small on-chip control store and a main control store. A small on-chip control store keeps the first two control words of each &mgr;-sequence. A large main control store contains the remaining control words of each &mgr;-sequence. The small control store is implemented so that there is no need to include an extra pip ...

Results 1 - 20 of 200 Result page: 1 2 3 4 5 6 7 8 9 10 next